Project 5 Report

Yaming Huang and Shuyi Zhang
College of Computer and Information Science
Northeastern University

1 Introduction

A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centers across the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance.[2]

In this project, we are given eight replica servers and one origin server to build our own simple CDN. These replica servers are located around the world. We will try to build our own simple CDN with better performances than two baselines: origin (a single server in a single EC2 location) and random server selection.

First, we use DNS redirection to send clients to the replica server with the fastest response time. Second, we write a simple Web server that returns content requested by clients. Third, we implement a system that determine the best replica server. We will introduce the general approach in Section 2. In Section 3 and 4, we will give the details about DNS server and Web server.

2 Approach

CDNs consist of:

1. a large number of servers geographically distributed worldwide;
2. a system that maps clients to "good" replica servers;
3. a system that determines those mappings.

In this project, we have one origin server and eight replica servers that are located in different cities around the world.

ec2-54-85-79-138.compute-1.amazonaws.com	Origin server
ec2-54-84-248-26.compute-1.amazonaws.com	N. Virginia
ec2-54-186-185-27.us-west-2.compute.amazonaws.com	Oregon
ec2-54-215-216-108.us-west-1.compute.amazonaws.com	N. California
ec2-54-72-143-213.eu-west-1.compute.amazonaws.com	Ireland
ec2-54-255-143-38.ap-southeast-1.compute.amazonaws.com	Singapore
ec2-54-199-204-174.ap-northeast-1.compute.amazonaws.com	Tokyo
ec2-54-206-102-208.ap-southeast-2.compute.amazonaws.com	Sydney
ec2-54-207-73-134.sa-east-1.compute.amazonaws.com	Sao Paulo

In order to map clients to “good” replica servers, we use DNS server redirection to send clients to the replica server with the fastest response time. To determine those mappings, we use active measurement and geolocation.

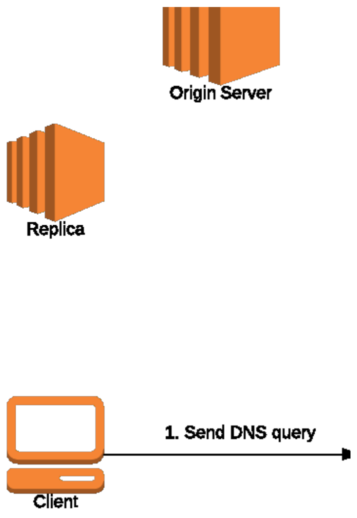

Figure 1. Client send DNS query

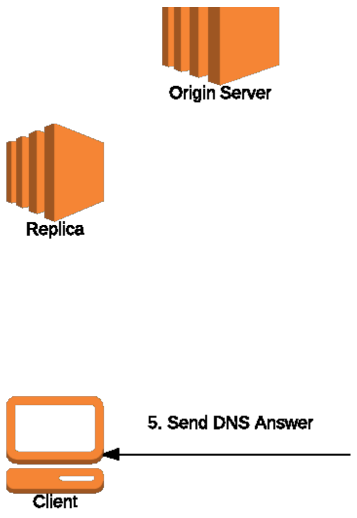

Figure 2. Send DNS response

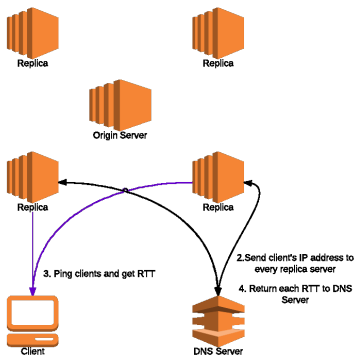
Replica
Replica
Origin Server
Replica
Replica
2.Send client's IP address to every replica server
3. Ping clients and get RTT
4. Return each RTT to DNS Server
Client
DNS Server

Figure 3. Mapping

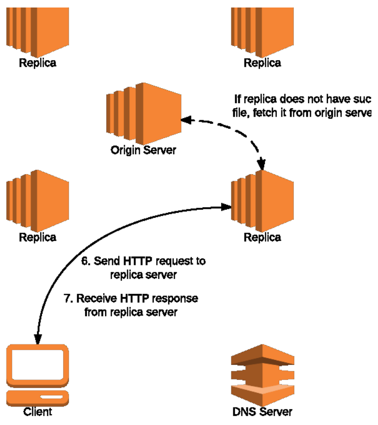
Replica
Replica
If replica does not have suc file, fetch it from origin serve
Origin Server
Replica
Replica
6. Send HTTP request to replica server
7. Receive HTTP response from replica server
Client
DNS Server

Figure 4. Client download files from replica server

The main steps of our CND working process:

1. As Figure 1 showed, firstly, the client will send a DNS query to our DNS server.
2. Then DNS server will send client’s IP address to every replica servers(In Figure 3).
3. Replica servers will try to ping the client and get RTT[1] between the client and replica servers(In Figure 3).
4. Replica servers send their RTT back to the DNS Server(In Figure 3).
5. DNS server sort RTT and select the replica with shorted RTT, then send the IP address of the best replica server(In Figure 2).
6. Client will access the given IP address and download files. If replica servers do not have the required files, they will fetch files from the origin server.(In Figure 4)

1. RTT: Round-Trip Time

3 DNS Server

3.1 DNS

The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates easily memorized domain names to the numerical IP addresses needed for the purpose of locating computer services and devices worldwide.[3]

In step 1, we need to parse a given DNS query. In step 5, we need to send a DNS response with a specific IP address. DNS primarily uses User Datagram Protocol (UDP) to serve requests. DNS queries consist of a single UDP request from the client followed by a single UDP reply from the server. So we build a UDP server using `SocketServer.UDPServer`[2] and write a program to parse and create DNS packet.

3.2 Mapping

When determining the replica server to use for each client request, we should determine which one will give the best time-to-completion (TTC) for downloading a Web page. This can depend on the network latency, loss, available bandwidth and load on the server. In this project, we use two methods to find the best replica server.

3.2.1 Active measurement

We use the following `scamper` command to measure latency: `scamper -c ‘ping -c 1’ -i [client IP]`.

However, active measurement has some limitations. For example, `scamper` command will not work if some servers or gateways block ICMP.

3.2.2 Geolocation

Since active measurement has some limitations, we will use IP geolocations to select the closest server to client when active measurement does not work.

We use IP location service of IPInfoDB.com[3]. After registering an account, we get an api key. We can do a query for the location of a given IP address by fetch such url: `http://api.ipinfodb.com/v3/ip-city/?key=<api_key>&ip=<ip address>`. The query result contains the latitude and longitude of the location of the given IP address. It is easy to calculate the distance between two location. So we can find the closest replica server to the client.

4 Web Server

4.1 HTTP Server

We build our HTTP server using `BaseHTTPServer.HTTPServer`[4]. The library has already implemented most of functions of HTTP server. We overwrite `do_GET` method, so HTTP server can download files from origin server if such files does not exist.

4.2 Cache Management

Since every replica only have 10 MB disk quota, we have to implement cache replacement algorithm. Least Recently Used (LRU) discards the least recently used items first. This algorithm requires keeping track of what was used when, which is expensive if one wants to make sure the algorithm always discards the least recently used item.[1]

2. `https://docs.python.org/2.7/library/socketserver.html#socketserver-udpserver-example`
3. http://ipinfodb.com/ip_location_api.php
4. `https://docs.python.org/2/library/basehttpserver.html`

In this project, we use a stack to implement LRU. Every time a page is accessed, push this page into the stack. When disk is full, delete the page in the bottom of the stack.

Reference

[1] Wikipedia. Cache algorithms — Wikipedia, the free encyclopedia. 2014. [Online; accessed 20-April-2014].

[2] Wikipedia. Content delivery network — Wikipedia, the free encyclopedia. 2014. [Online; accessed 20-April-2014].

[3] Wikipedia. Domain name system — Wikipedia, the free encyclopedia. 2014. [Online; accessed 20-April-2014].